第32卷 第20期 计 算 机 工 程 2006年10月
Vol.32 № 20 Computer Engineering October 2006
·安全技术· 文章编号：1000—3428(2006)20—0143—04 文献标识码：A 中图分类号：TP393.08

基于不定长系统调用序列模式的入侵检测方法

王福宏，彭勤科，李乃捷
(西安交通大学电子与信息工程学院，西安 710049)

摘 要：提出了一种不定长序列模式的寻找算法，目标是从训练序列中找出一组基本的、相对独立的不定长序列模式。并在模式集的更新过程中自动定义了模式间的前后次序关系，以此构建了一个描述进程执行模式的DFA。针对已有基于不定长序列模式的模式匹配算法需要向前预测若干个系统调用号的缺点，文章设计了一个更好的模式匹配算法。实验结果表明，算法在模式寻找过程中是稳定的，并在保持一组规模很小的模式集的情况下，取得了很低的误报率和漏报率。

关键词：入侵检测；系统调用；模式匹配；不定长序列模式；误报率

Intrusion Detection Using Variable-length System Calls Patterns

WANG Fuhong, PENG Qinke, LI Naijie
(School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049)

【Abstract】A novel simple technique to build a table of variable-length patterns from training system call sequences is presented, aiming to find out a set of basic and relatively independent variable-length patterns. Also, the method finds out all possible relationship between variable-length patterns, and thereby generates an exact DFA representation of the program. Using the data sets from the university of New Mexico, the schema is evaluated by several targets—sizes of variable-length patterns, false positives and false negatives. The experimental results indicate that the algorithms generate a relative small set of patterns, and get very low false positives and false negatives.

【Key words】 Intrusion detection; System call; Pattern match; Variable-length patterns; False positives

通过分析进程执行中产生的系统调用序列可抽取一组描述进程行为的模式，并根据该模式与进程实时产生的系统调用序列的匹配情况，可以检测出入侵行为。Forrest等首先提出了一种利用系统调用序列进行入侵检测的方法[1]，其主要是利用一组定长的短序列模式来描述进程的正常运行模型。该方法的主要局限性是很难合理选择短序列长度，一般来说，短序列模式越长，入侵检测倾向于在系统调用序列中捕捉越多的异常，但同时模式集规模也会越大，算法执行效率会降低，且误报率也会增加[3]；而序列模式越短则越难捕捉到异常。Lee等人利用条件熵模型来求最优短序列的长度[4]，然而，由于系统调用序列收集过程跨越了很长的时间段，在整个数据集上采用条件熵模型不一定合理。作者编写的脚本测试也表明，这种方法得到的最优短序列长度应用到Forrest等在文献[1]中给出的stide算法中时，算法的检测能力并没有得到明显改善。由于程序具有结构性特征，因此采用不定长序列模式来描述进程的正常行为模式会更加合理。但是，在系统调用序列中寻找不定长序列模式是非常困难的。目前所采用的寻找算法都较复杂[3,5,7]，且这些算法除了要求大量的训练集以外，很难对已产生模式集增量更新。以Wespi等的方法为例[7]，如果已有的最大模式集不能够覆盖新的训练样本，则无法对现有的模式集增量更新。且在模式匹配过程中，没有考虑模式间的次序关系，从而会漏掉某些进程在执行过程中执行路径发生变化引发的异常。

1 算法描述

1.1 基本定义

(1) Σ：是程序所有可能执行的系统调用号的集合。

(2)序列：是 Σ 中的系统调用号所组成的任何有穷序列，如序列 $s=\{a_1,a_2,...,a_n\}$，$(a_i\in\Sigma,1\le i\le n)$。

(3) $|s|$：序列 s 的长度，表示序列 s 中的系统调用号个数。

(4) Σ^*：是指 Σ 上所有序列的全体。如 $\Sigma^*=\{s_1,s_2,s_3,\ldots\}$，$s_i$ 为定义在 Σ 上的一条序列。

(5) s^n：表示序列 s 自身连接 n 次得到的序列，n 表示序列 s 的重复度。

(6) $s_1\cup s_2$：表示序列 s_1 与序列 s_2 合并。对于定义在 Σ 上的两条序列 $s_1=\{a_1,a_2,...,a_n\}$，$s_2=\{b_1,b_2,...,b_m\}$，则

$p_1\cup p_2=\{a_1,a_2,...,a_n,b_1,b_2,...,b_m\}$。

1.2 模式抽取

定义 1 给定一组训练集 $T=\{T_1,T_2,...,T_m\}$，T 中第 i 条序列为 T_i，$T_i\in\Sigma^*$，$1\le i\le m$，T_i 中第 j 个系统调用号记为 t_{ij}，$t_{ij}\in\Sigma$，$1\le j\le|T_i|$。定义 Σ 上的一条序列 $p=\{a_1,a_2,...,a_n\}$，$(a_i\in\Sigma,1\le i\le n)$，如果满足下面 3 个条件:

(1)序列 p 在训练集 T 上至少出现一次；

(2)序列的长度 $|p|\ge 2$；

(3) $a_i\ne a_j$，$1\le i,j\le n$。

则称序列 p 为一个模式。在一个句子 T_i 中，如果相同的字符 a 连续重复出现，则将最小模式定义为 $p=\{a,a\}$。

基金项目：国家自然科学基金资助项目(60373107)；国家“863”计划基金资助项目(2003AA142060)

作者简介：王福宏(1980－)，男，硕士生，主研方向：集群环境下的基于符号序列的入侵检测系统，机器学习，计算机集群计算；彭勤科，教授；李乃捷，硕士生

收稿日期：2005-12-31 **E-mail：**qkpeng@mail.xjtu.edu.cn

定义 1 给出了一种自底向上的模式抽取算法，该算法对输入序列 s 自左向右扫描，并将输入的系统调用序列逐个移进一个队列中，边移进边分析；一旦队列中序列符合定义 1，则将其中系统调用号移出到模式 p 中，重复上述过程，直到对输入句子 s 扫描结束，下面举例说明。

例 1 给定由 sendmail 客户进程正常执行产生的 3 条系统调用序列组成的训练集 $T=\{T_1, T_2, T_3\}$，其中

T_1 ={open, read, mmap, mmap, open, getrlimit, mmap, open, getrlimit, mmap, open, read, getrlimit, mmap, close};

T_2 = {open, read, mmap, mmap, open, getrlimit, mmap, open, read, getrlimit, open, close};

T_3 = {open, read, mmap, mmap, open, getrlimit, mmap, open, read, open, getrlimit, close};

根据表 1 提供的对应关系，将上述训练集 T 转换为符号序列：{ABCCADCABDCE, ABCCADCABDAE, ABCCADCABADE}。

表 1 系统调用函数名与系统调用号对照表

系统调用函数名	open	read	mmap	getrlimit	close	…
系统调用号	A	B	C	D	E	…

基于定义 1，在序列 T_1 中抽取模式的过程如图 1 所示。最后在整个训练集 T 上模式抽取的输出结果为模式集合 $\{P_1, P_2, P_3\}$，其中

P_1 = {ABC, (C AD)^2, C ABD, CE},

P_2 = {ABC, C AD, C ABD, AE},

P_3 = {ABC,C AD, CAB, ADE}

如例 1 所示，根据定义 1，对于训练集 T 中任何一条序列 T_i，可以分解为

$$T_i = p_{i_1}^{l_{i_1}} p_{i_2}^{l_{i_2}} \cdots p_{i_{n_i}}^{l_{i_{n_i}}}\ ,\ \sum_{j=1}^{n_i}(l_{i_j}\times|p_{i_j}|)=|T_i|\ ,\ l_{i_j}\ge 1\ ,\ 1\le j\le n_i$$

n_i 表示序列 T_i 上分割出的独立的模式数。在序列 T_i 进行模式抽取的输出结果是 $P_i=\{p_{i_1}, p_{i_2}, ..., p_{i_{n_i}}\}$。算法的复杂度是 $O(|T_i|)$。

该算法的输出结果是唯一的，这样就解决了Kosoresow的方法寻找宏时所遇的宏的个数和长度不确定问题[5]。

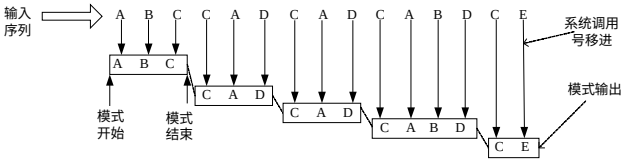

图 1 模式抽取过程

1.3 更新模式集

定义 2 设 $\Phi=(P,F,N,U)$ 是一个定义在 Σ 和训练集 T 上的模式集，其中第 i 条记录表示为 $\Phi_i=(P_i, f_i, N_i, U_i)$。其中: P_i 是不定长序列模式 p_i，$P_i\in P$；f_i 是 p_i 出现的频数，$f_i>0$，$f_i\in F$；N_i 是 p_i 之后所有可能的模式编号，$N_i\in N$，且 N_i 可以为 ϕ，表示在 p_i 之后没有其它模式出现；U_i 是 p_i 之前所有可能的模式编号，$U_i\in U$，且 U_i 可以为 ϕ，表示在模式 p_i 之前没有其它模式出现。

更新模式集的过程就是根据定义 2，将训练序列 T 上产生的模式集合 $\{P_1, P_2, P_3\}$ 插入到模式集 Φ 中。更新过程分为两个步骤:

(1)对于训练序列 T_i 上产生的模式集 $\{p_{i_1}, p_{i_2}, ..., p_{i_{n_i}}\}$，查询每个模式 p_{i_j} 在集合 P 中的编号 k，并将相应的 f_k 值加 l_{i_j}；如果不存在，则将 p_{i_j} 追加到集合 P 中，并记录对应的编号 k，将相应的 f_k 值设置为 l_{i_j}；

(2)扫描整个模式集合 $\{p_{i_1}, p_{i_2}, ..., p_{i_{n_i}}\}$，将 $p_{i_{(j-1)}}$ 的编号添加到模式 p_{i_j} 的编号 k 所对应的集合 U_k 中；将 $p_{i_{(j+1)}}$ 的编号添加到模式 p_{i_j} 的编号所对应的集合 N_k 中；如 $l_{i_j}>1$，则将 p_{i_j} 的编号添加到模式 p_{i_j} 的编号所对应的集合 U_k 和 N_k。有两种特殊情况: p_{i_1} 的编号 k 所对应的集合 U_k 中加入 ε，表示是序列 T_i 开始的模式； $p_{i_{n_i}}$ 的编号 k 所对应的集合 N_k 中加入 ε，表示是训练序列 T_i 结束的模式。

设初始模式集 $\Phi=\phi$，将 T 上的原始模式 $\{P_1, P_2, P_3\}$ 插入最终模式集 $\Phi(P,F,N,U)$ 的过程如图 2，ABC(1,2)含义为：ABC 表示模式、1 表示模式编号、2 表示模式的频数，图中箭头表示了模式之间的先后次序关系。

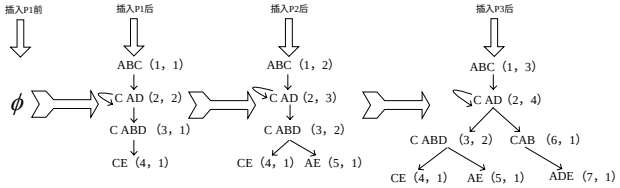

图 2 模式集更新过程图示

实验表明（如图 3），随着训练数据的增加，模式集的规模趋向于稳定。

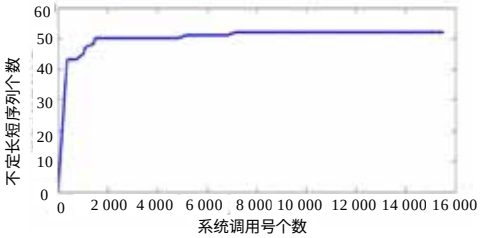

图 3 sendmail 客户进程模式集的创建过程

可以证明模式集更新算法的复杂度是 $O(n_i\times|P|)$，n_i 为序列 T_i 的分段数，P 为 Φ 中更新前的模式集合。

1.4 精简模式集

由于较长的序列模式更有可能捕捉到入侵行为，因此要对模式进行合并。

定义 3 如果集合 P 中的模式 p_i 和 p_{i+1} 满足下面 3 个条件：

(1) $f_i=f_{i+1}, 1\le i<|F|$；

(2) $N_i=\{i+1\}$；

(3) $U_{i+1}=\{i\}$

则合并模式 p_i 和 p_{i+1}。合并过程分以下 3 步:

(1) $p_i=p_i\cup p_{i+1}$；

(2) $N_i=N_{i+1}$；

(3)从模式集合 P 删除记录 p_{i+1}。

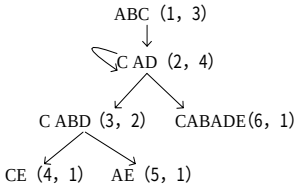

图 4 精简后的模式集图示

精简图 2 的模式集，结果如图 4 所示。图 3 中 CERT 的 sendmail 进程的模式精简后剩下 28 条，见图 5。

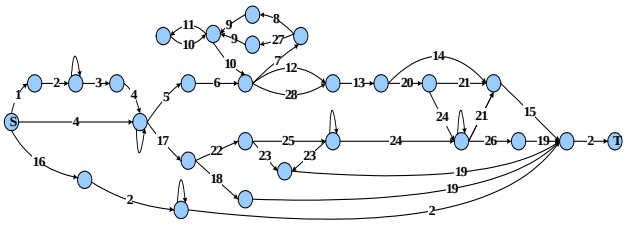

图 5　sendmail 客户进程的 DFA 匹配算法

1.5 构造 DFA

DFA是描述离散系统的有效工具。进程可以看作为一个典型的离散系统[5]，图 5 是对CERT数据集中的sendmail进程分析后得到的模式间状态转移图，其中S表示开始节点，T表示终止节点，弧线对应不定长序列模式。图 6 中清晰显示了用sendmail发送信息时执行的 3 类客户进程，并反映出sendmail客户程序的一些结构特征。

1: 4 2 66 66 4 138 66 5 23 45 4 27 66 5	**13:** 50 27 2 5 17 4 14 50 27 2
4 2 66 66 5 4 2 66 66 5 5 85	**14:** 50 27 2 3 5 93 88 112 19 128
2: 5 5	**15:** 112 19 128 4 4 14 50 27 3 5 9 124 1 112
3: 5 4 59 155 4 50 27 2 5 105 104 104 106	4 5 5
4: 105 104 104 106	**16:** 1 5 5
5: 105 104 104 106 56 19 155 83	**17:** 105 104 104 106 5 4 5 5 40 40 41 105 104
6: 19 4 93 94 5 112	104 106 61 5 85 50 27 18
7: 19 93	**18:** 50 27 2 2 3 5 6 6 112 112 19 128 9 9
8: 19 100 50 128 85 89 121 5 5 4	**19:** 5 9 9 5 112 4
9: 50 27 2 19 4 93 94 5 112	**20:** 50 27 2 3
10: 19 93 100 50 128 89 121 5 5	**21:** 2 3 5 93 88 112 19 128
11: 5 93 50 112 4 2 5 23 4 50 27 2 2 2 54	**22:** 50 27 2 3 3
4 50 27 2 2 5 11 17 4 4 27 17 112 19 19 32	**23:** 2 2 3 5 6 6 112 112 19 128 9 9
32 7 5 8 9 8 4 14 112 19 112 4 2 5 112 112	**24:** 2 3
50 50 50 27 4 27 88 167 167 17 5 4 50 27	**25:** 2 3 3
2 5 4 18 2 5 4 50 27 2 5 23 23 4 50 27 2	**26:** 2 3 5 6 6 112 112 19 128 9 9
5 18 2 19 4 93 94 5 112	**27:** 19 100 100 50 128 85 89 121 5 5 4
12: 19 93 100 50 128 89 121 5 5 4	**28:** 19 93 100 100 50 128 89 121 5 5 4

图 6　精简后的 sendmail 客户进程模式列表

2 匹配算法

定义 4 候选模式集 C 是定义在模式集 Φ 上的一个三元组 (P,SP,CP)，候选模式集中第 i 个元组表示为 (p_i,sp_i,cp_i)。其中，p_i 表示第 i 条当前参与匹配的模式的编号；sp_i 表示模式 p_i 开始匹配的位置；cp_i 表示模式 p_i 当前匹配的位置；(sp_i,cp_i) 共同构成了候选模式集中模式 p_i 上的位置信息。

选择合理的异常评估尺度可以有效地提高入侵检测能力和检测质量。目前有许多种评估尺度[6,7]，Kosoresow等提出计算序列中一个长度为 l 的固定区域 LF 内发生的误匹配的个数 LFC [5]，并预先设定一个阈值 τ，当 $LFC>\tau$ 时进行报警。由于异常倾向于在系统调用序列局部集中出现[2]，因此本文的匹配算法采用了这种评估尺度，在本文后续的实验中，指定 $l=20$。详细的在线算法如下：

(1)初始化参数：模式候选集 $C=\phi$；长度为 l 的固定区域内的误匹配计数值 $LFC=0$；设定阈值 τ；当前位置 $cur\_pos=0$，上次成功匹配位置 $ls\_pos=0$；

(2) 读入一个系统调用号 sc，序列当前位置 $cur\_pos=cur\_pos+1$；

(3)如果候选集 $C=\phi$ 或者 C 中没有模式能与 sc 匹配，则 $C=search(\Phi,sc)$，即从模式集 Φ 中找出所有以 sc 开始的模式编号存入候选集 C；否则更新 C，除去不能与 sc 匹配模式；更新 C 中剩余模式的当前位置信息 CP，如对于第 i 个元组 (p_i,sp_i,cp_i)，令 $cp_i=cp_i+1$；

(4)如果 C 中某一模式 p 被完整匹配后，则从该模式的 sp 到 cp 所对应的序列位置全部标为 0；假设该模式在 Φ 中所对应的编号为 k，则将 N_k 中的模式编号信息全部加入候选集 C 中；更新 C 中剩余模式的初始位置信息 SP，如对于第 i 个元组 (p_i,sp_i,cp_i)，令 $sp_i=cp$；上次成功匹配位置 $ls\_pos=cur\_pos$；

(5)如 C 没有模式与当前读入的系统调用号 sc 匹配，则从上一次模式匹配成功的位置到 cur_pos 全部标为 1；

(6) LF 向前滑动一个位置，计算该固定区域内的误匹配个数 LFC。如果 $LFC>\tau$，则进行报警；

(7)返回步骤(2)。

图 7 为一张典型的误匹配分布图。图中数据来自新墨西哥大学提供的 ps 进程序列数据，攻击序列采用 Trojan 木马程序。深浅相隔的线表示属于不同的进程。

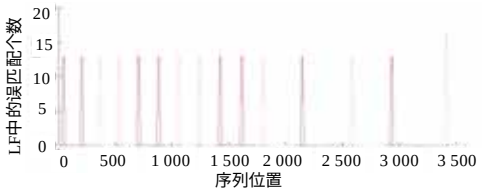

图 7　系统调用误匹配分布

图 7 中误匹配在序列中的分布符合 Trojan 木马程序的特点。可以看出，算法可以有效地检测出这类入侵。

Forrest[2,6]和Wespi[7]模式匹配的共同点在于用当前序列模式与模式库的匹配情况来检测入侵。然而，对于有些入侵序列，采用它们的方法是检测不到的。如一条入侵序列为{ABCCABDADCAE}，将其分割为{ABC－CABD－ADC－AE}。可以看出模式{ABC,CABD,ADC,AE}都在模式集P中，所以用Forrest和Wespi的方法检测不到模式CABD和ADC颠倒所发生的异常。而用本文的匹配方法，由于模式集P中不存在从模式CABD转移到模式ADC，因此这种模式间的转移行为被视为入侵行为。

3 实验

3.1 异常检测

Forrest[1]和Lee[3]利用在整条序列中误匹配序列所占的比例来区别正常和异常序列。为了测试算法对序列数据中异常的检测能力，本文做了Lee[3]中的异常检测的对比实验。实验结果如表 2 所示。

表 2　异常检测结果的比较

攻击序列	异常度×100		
	Forrest[1]	Lee[3]	本文算法
sscp-1	5.2	13.5	25.8
sscp-2	5.2	13.6	25.8
sscp-3	5.2	13.6	25.8
syslog-r-1	5.1	11.5	42.5
syslog-r-2	1.7	8.4	49.4
syslog-l-1	4.0	6.1	41.9
syslog-l-2	5.3	8.0	43.5
decode-1	0.3	3.9	27.2
decode-2	0.3	4.2	27.2
forwardloop	2.3	−	24.9
sm565a	0.6	8.1	47.9
sm5x	2.7	8.2	35.7
Normal	0	0.6	0

从表 2 可见，本文算法异常度的阈值选择范围大。由于在攻击序列中插入大量系统调用，可以稀释给定区域内的误匹配次数，因此本文算法利用不定长序列模式与进程执行所产生的系统调用序列进行匹配。

3.2 误报率和漏报率

误报率和漏报率是评价入侵检测系统的两个重要指标。由于异常都倾向于在系统调用序列中局部集中出现[2]，因此本文为LFC设定了一个阈值 τ，当LFC值超过 τ 时，便认为

该序列异常。显然，LFC阈值 τ 的选取非常关键，本文采用Warrender中的办法[2]，阈值 τ 取实报率达到95%以上时的最大LFC值。表 3 数据来自新墨西哥大学和DARPA数据，Forrest[1]的算法采用的滑动窗口长度为 6，实验结果如表 4。

表 3 实验数据

进程	攻击序列	正常数据 所有数据		正常数据 训练数据		正常数据 测试数据	
	进程数	进程数	调用号数	进程数	调用号数	进程数	调用号数
MIT lpr	1001	2 703	2 926 304	415	568 733	1 645	1 553 768
UNM lpr	1001	1231	434 303	250	86 652	981	347 651
login	8	9	8 887	9	8 887	–	–
ps	21	24	6 144	24	6 144	–	–
stide	105	13 726	15 618 237	200	246 750	13 526	15 185 927
samba	4	12	40 834	5	15 396	7	25 438
httpd	33	18	20 502	6	7 001	12	13501

表 4 实验结果

进程	本文算法 模式集规模	误报率	实报率	阈值	Forrest[1] 模式集规模	误报率	实报率	阈值
MIT lpr	109	2.5%	100%	2	452	0.43%	95.2%	6
UNM lpr	60	0%	100%	X	280	0%	100%	X
login	65	–	75%	X	375	–	75%	14
ps	34	–	100%	9	173	–	100%	4
stide	30	0.08%	95.24%	12	150	0.25%	95.24%	4
samba	81	0%	100%	19	448	0%	100%	5
httpd	10	0%	100%	X	40	0%	100%	X

表 4 的阈值一列中，数字表示实报率达到 95%以上的最大阈值，X 表示可取任意阈值。可以看出，对多数进程，至少有一个阈值可以使得实报率高于 95%，而误报率相比是很低的。另外，本文算法的分析速度也是很快的。在训练过程中，每秒平均可分析 23 000 多个调用号（Forrest 的训练方法只有 1 250 个/s）。有时候，尽管模式集规模比较大，但由于模式之间的转移关系比较简单，即 N 中没有很长的数组，算法分析速度仍然是很快的。

4 总结

基于程序的结构特征，本文提出了一种快速的不定长序列模式寻找算法，并首次将模式间次序关系引入到了匹配算法中。实验结果表明，本文算法相比其它基于系统调用短序列的算法，可在入侵序列中捕捉到更多异常信息。并在保持一组规模很小的模式集的情况下，取得了很低的误报率和漏报率。

参考文献

1 Forrest S, Hofmeyr S A, Somayaji A, et al. A Sense of Self for Unix Processes[C]. Proceedings of the IEEE Symposium on Security and Privacy, 1996:120-128.

2 Warrender C, Forrest S, Pearlmutter B. Detecting Intrusions Using System Calls: Alternative Data Models[C]. Proceedings of the IEEE Symposium on Security and Privacy, 1999: 133-145.

3 Lee W, Stolfo S J. Data Mining Approaches for Intrusion Detection[C]. Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas, 1998.

4 Eskin E, Lee W, Stolfo S J. Modeling System Calls for Intrusion Detection with Dynamic Window Sizes[C]. Proceedings of DARPA Information Survivability Conference & Exposition II, 2001.

5 Kosoresow A P, Hofmeyer S A. Intrusion Detection via System Call Traces[J]. IEEE Software, 1997, 14(5): 35-42.

6 Hofmeyr S A, Forrest S, Somayaji A. Intrusion Detection Using Sequences of System Calls[J]. Journal of Computer Security, 1998, 6(3): 151-180.

7 Wespi A, Dacier M, Debar H. Intrusion Detection Using Variable-length Audit Trail Patterns[C]. Proceedings of Workshop on Recent Advances in Intrusion Detection, Toulouse, France, 2000.

(上接第 85 页)

a=1 ⟶ d=0，一致程度为 1。
a =2 且 b =0 ⟶ d =1，一致程度为 1。
a =2 且 c=0 ⟶ d =0，一致程度为 1/2。
a =2 且 b =1 ⟶ d =1，一致程度为 1/2。
a =3 且 b =0 ⟶ d =0，一致程度为 1。
b=1 且 c=0 ⟶ d =0，一致程度为 1/2。
b =1 且 c=0 ⟶ d =1，一致程度为 1/2。
a =3 且 c=1 ⟶ d =1，一致程度为 1。

表 3 属性值约简

U	a	b	c	d
1	1	×	×	0
2	1	×	×	0
3	2	0	×	1
4	2	×	0	0
5	2	1	×	1
6	3	0	×	0
7	×	1	0	0
8	×	1	0	1
9	3	×	1	1

通过算法对决策表进行属性值的约简，即能够得到更为简洁的决策规则，但是却只需要少量的数据处理。同其它规则相比较，获取方法将节省大量的资源和时间，是值得考虑的方法。

5 结束语

本文提出了基于粗糙集的网络用户访问规则的获取新方法，该算法的优势就在于在处理少量数据的基础上就能得到更为简洁的决策规则。实例分析验证了该算法的可行性，并能很好地处理一致决策表和不一致决策表。需要进一步研究的是取得更为有效的特征属性及其数值。

参考文献

1 Chen M S, Park J S, Yu P S. Efficient Data Mining for Path Traversal Patterns[J]. IEEE Trans. on Knowledge and Data Engineering, 1998, 10(2): 209-221.

2 Han J, Kamber M. 数据挖掘: 概念与技术[M]. 北京: 机械工业出版社, 2001.

3 张文修, 吴伟志. 粗糙集理论与方法[M]. 北京: 科学出版社, 2001.

4 Pawlak Z. Rough Set[J]. International Journal of Computer and Information Science, 1982, 11(5): 341-356.

5 王国胤. Rough 集理论与知识获取[M]. 西安: 西安交通大学出版社, 2001.

6 刘少辉, 盛秋戬, 吴斌等. Rough 集高效算法的研究[J]. 计算机学报, 2003, 26(5): 524-529.